Equalizer Lab

Lab 6

Section 7

Submitted By:

Benjamin Muslic

Submission Date:

11/3/2021

Lab problem

The week long lab seeks to test the user into making a program that outputs letters as he/she tilts the DUALSHOCK 4 controller at a specific side. The user can also utilize extra credit by making the source code where it outputs a different side of turning at the press of a button.

Analysis

In order to do this lab the user is required to use experience from past labs and textbook exercises to be prepared enough to make a source code that succeeds in the lab problem. The code was provided with a skeletal start to help the user understand the program and what they were going to do to successfully output the program. The user needed to define multiple functions and was only allowed to use one scanf statement making a further challenge.

Design

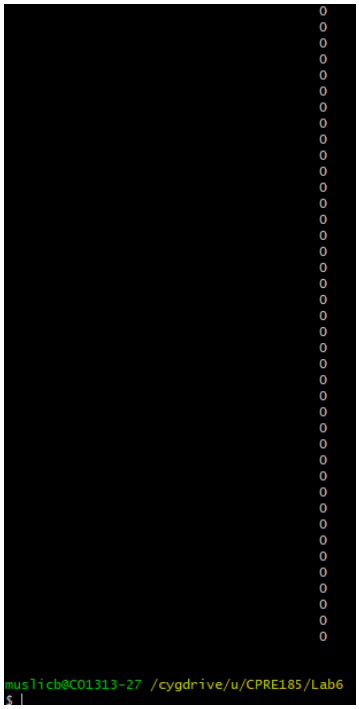
```c
// 185 lab6.c
//
// This is the outline for your program
// Please implement the functions given by the prototypes below and
// complete the main function to make the program complete.
// You must implement the functions which are prototyped below exactly
//  as they are requested.

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define PI 3.14159265358

//NO GLOBAL VARIABLES ALLOWED

printf("Benjamin Muslic");
//PRE: Arguments must point to double variables or int variables as appropriate
//This function scans a line of DS4 data, and returns
//  True when the square button is pressed
//  False Otherwise
//This function is the ONLY place scanf is allowed to be used
//POST: it modifies its arguments to return values read from the input line.
int read_line(double*g_x, double*g_y, double*g_z,int*time,int*Button_T, int*Button_X, int*Button_S,int*Button_C);


// PRE: -1.0 <= x_mag <= 1.0
// This function computes the roll of the DS4 in radians
// if x_mag outside of -1 to 1, treat it as if it were -1 or 1
// POST: -PI/2 <= return value <= PI/2
double roll(double x_mag);

// PRE: -1.0 <= y_mag <= 1.0
// This function computes the pitch of the DS4 in radians
// if y_mag outside of -1 to 1, treat it as if it were -1 or 1
// POST: -PI/2 <= return value <= PI/2
double pitch(double y_mag);


// PRE: -PI/2 <= rad <= PI/2
// This function scales the roll value to fit on the screen
// POST: -39 <= return value <= 39
int scaleRadsForScreen(double rad);

// PRE: num >= 0
// This function prints the character use to the screen num times
// This function is the ONLY place printf is allowed to be used
// POST: nothing is returned, but use has been printed num times
void print_chars(int num, char use);

//PRE: -39 <= number <=39
// Uses print_chars to graph a number from -39 to 39 on the screen.
// You may assume that the screen is 80 characters wide.
void graph_line(int number, int mode);

int main()
{
    double x, y, z;                          // magnitude values of x, y, and z
    int b_Triangle, b_X, b_Square, b_Circle, t, b4press;
    double roll_rad, pitch_rad;                // value of the roll measured in radians
    int scaled_value,mode=0;               // value of the roll adjusted to fit screen display


    // ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g -b | ./lab6


    //insert any beginning needed code here
    do
    {
        // Get line of input
    read_line(&x, &y, &z, &t, &b_Triangle, &b_X, &b_Square, &b_Circle);
        // calculate roll and pitch.  Use the buttons to set the condition for roll and pitch
    roll_rad = roll(x);
    pitch_rad = pitch(z);
        // switch between roll and pitch(up vs. down button)
    if (b_Triangle==1 && b4press==0){
        if(mode==0) mode=1;
        else if (mode==1) mode=0;
    }
    b4press=b_Triangle;
        // Scale your output value
    if (mode==0){
        scaled_value = scaleRadsForScreen(roll_rad);
    }
    if (mode==1){
        scaled_value=scaleRadsForScreen(pitch_rad);
    }
```

```c
// You may assume that the screen is 80 characters wide.
void graph_line(int number, int mode);

int main()
{
    double x, y, z;                             // magnitude values of x, y, and z
    int b_Triangle, b_X, b_Square, b_Circle, t, b4press;
    double roll_rad, pitch_rad;                 // value of the roll measured in radians
    int scaled_value,mode=0;                // value of the roll adjusted to fit screen display


    // ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g -b | ./lab6


    //insert any beginning needed code here
    do
    {
        // Get line of input
    read_line(&x, &y, &z, &t, &b_Triangle, &b_X, &b_Square, &b_Circle);
        // calculate roll and pitch.  Use the buttons to set the condition for roll and pitch
    roll_rad = roll(x);
    pitch_rad = pitch(z);
        // switch between roll and pitch(up vs. down button)
    if (b_Triangle==1 && b4press==0){
        if(mode==0) mode=1;
        else if (mode==1) mode=0;
    }
    b4press=b_Triangle;
        // Scale your output value
    if (mode==0){
        scaled_value = scaleRadsForScreen(roll_rad);
    }
    if (mode==1){
        scaled_value=scaleRadsForScreen(pitch_rad);
    }

        // Output your graph line
    graph_line(scaled_value, mode);

        fflush(stdout);
    } while (b_Square==0);
    return 0;
}

int read_line(double* g_x, double* g_y, double* g_z, int* time, int* Button_T, int* Button_X, int* Button_S, int* Button_C)
{
    scanf("%d,%lf,%lf,%lf,%d,%d,%d,%d", time, g_x, g_y, g_z, Button_T, Button_C, Button_X, Button_S);

    if (*Button_S ==1){
        return 1;
    }
    else {
        return 0;

    }
}

double roll (double x_mag){
    double result;

    if (x_mag > 1.0) {
        x_mag = 1;
    }
    else if (x_mag < -1.0){
        x_mag = -1;
    }
    x_mag = asin(x_mag);
}

double pitch (double y_mag){
    double result;

    if (y_mag > 1.0) {
        y_mag = 1;
    }
    else if (y_mag < -1.0){
        y_mag = -1;
    }
    y_mag = asin(y_mag);
}

int scaleRadsForScreen(double rad){
    rad=rad*(78.0/PI);
    return rad;
}
```

```c
    if (*Button_S ==1){
        return 1;
    }
    else {
        return 0;

    }
}

double roll (double x_mag){
    double result;

    if (x_mag > 1.0) {
        x_mag = 1;
    }
    else if (x_mag < -1.0){
        x_mag = -1;
    }
    x_mag = asin(x_mag);
}

double pitch (double y_mag){
    double result;

    if (y_mag > 1.0) {
        y_mag = 1;
    }
    else if (y_mag < -1.0){
        y_mag = -1;
    }
    y_mag = asin(y_mag);
}

int scaleRadsForScreen(double rad){
    rad=rad*(78.0/PI);
    return rad;
}

void print_chars(int num, char use){
    int b;

    for (b = 0; b < num; b++){
        printf("%c", use);
    }
}
void graph_line(int number, int mode)
{
    number *= -1;

    if (mode==0)
    {
        if (number < 0)
        {
            print_chars(39 + number, ' ');
            print_chars(abs(number),'L');
        }
        if (number >= 0)
        {
            print_chars(39, ' ');
            if (number == 0)
            {
                print_chars(1, '0');
            }
            print_chars(number, 'R');
        }
        print_chars(1,10);
    }
    if (mode==1) {
        if (number < 0)
        {
            print_chars(39 + number, ' ');
            print_chars(abs(number),'F');
        }
        if (number >= 0)
        {
            print_chars(39, ' ');
            if (number == 0)
            {
                print_chars(1, '0');
            }
            print_chars(number, 'B');
        }
        print_chars(1,10);
    }

}
```

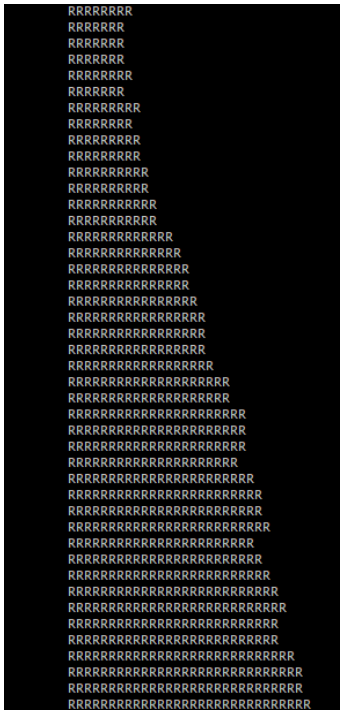
The above pictures shows the source code for this program. The program requires utilization of several functions and else statements and only one scanf statement. At the same time, to get letters repeating loops were needed to be used as well.

Testing

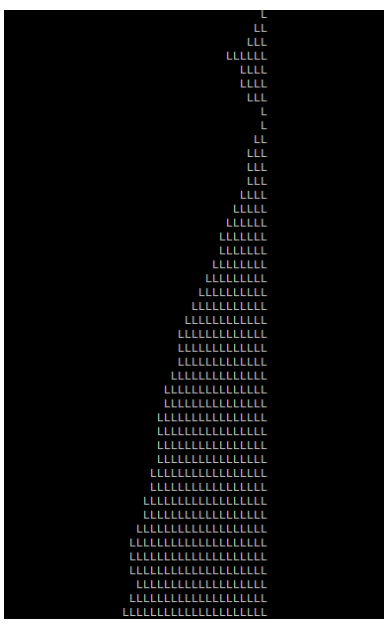


```
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0
                                       0

muslicb@CO1313-27 /cygdrive/u/CPRE185/Lab6
$ 
```

The output above shows the program in action when the controller is not being touched at all.

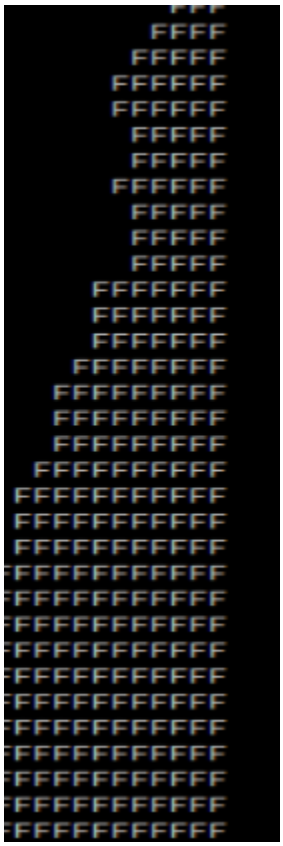


The output above shows the program in action when the controller is turned to the **RIGHT**

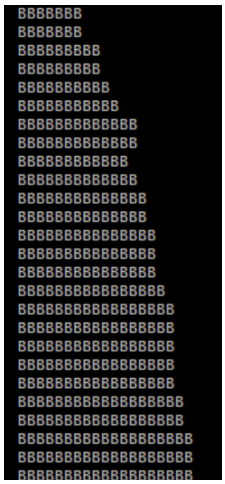


The output above shows the program in action when the controller is turned to the **LEFT.**

Bonus



The output above shows the program in action when the controller is tilted **FORWARD**.



The output above shows the program in action when the controller is tilted **BACWARD**

Analysis

How did you scale your values? Write an equation and justify it.

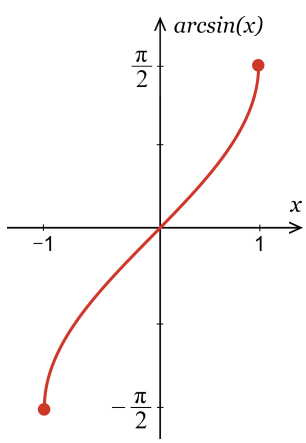
I scaled my values in relation to what was given to us for the lab. The values scaled from -39 to 39 so that the repeating letters (doesn't matter which L,R,F, or B) can fit on the screen and eventually stop to a certain point. The equation that was used to help achieve this is:

$$\frac{78}{\pi}$$

Where π is defined as “PI” in the source code as 3.141592653589.

How many degrees does each letter in your graph represent? This is the precision of your graph. As your experiment with the roll and pitch, what do notice about the graph's behavior near the limits of its values?

If one were to look at an arcsin graph (shown below) the scaled values are from -1 to 1 and in terms of Y it is from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ in radians so those would be the values but in radians. For the graph behavior the change is much faster as it reaches the limits.



Comments

Another interesting lab. It was cool to see how we can combine C skills and start to Understand how the dualshock 4 really works in real life. This was the closest experience I had to sort of playing a video game where I coded the controls. The fact that plenty of things what happens as I move the controller is fascinating. As always, I can't wait to see what the next lab offers.